**AFRL-IF-RS-TR-2004-136**
**Final Technical Report**
**May 2004**


# DETECTOR AND EXTRACTOR OF FILEPRINTS (DEF) PROTOTYPE

**Black River Systems Company, Inc.**

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

**AIR FORCE RESEARCH LABORATORY**
**INFORMATION DIRECTORATE**
**ROME RESEARCH SITE**
**ROME, NEW YORK**

**STINFO FINAL REPORT**

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-136 has been reviewed and is approved for publication

APPROVED: /s/
ANDREW J. NOGA
Project Engineer

FOR THE DIRECTOR: /s/
JOSEPH CAMERA, Chief
Information & Intelligence Exploitation Division
Information Directorate

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 074-0188*



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | MAY 2004 | **FINAL** **Mar 03 – Nov 03** |

**4. TITLE AND SUBTITLE**

DETECTOR AND EXTRACTOR OF FILEPRINTS (DEF) PROTOTYPE

**5. FUNDING NUMBERS**

C - F30602-03-C-0068
PE - 63789F
PR - STG3
TA - 03
WU - 04

**6. AUTHOR(S)**

Richard Henry

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Black River Systems Company, Inc.
162 Genesee Street
Utica NY 13502

**8. PERFORMING ORGANIZATION REPORT NUMBER**

N/A

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

AFRL/IFEC
32 Brooks Road
Rome NY 13441-4114

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

AFRLIF-RS-TR-2004-136

**11. SUPPLEMENTARY NOTES**

AFRL Project Engineer: Andrew J. Noga/IFEC/(315) 330-2270 Andrew.Noga@rl.af.mil

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** ***(Maximum 200 Words)***

The Detector and Extractor of Fileprints (DEF) algorithm is a technique for representing data sequences in the form of fileprints. It combines a periodogram-based process with the Adjustable Bandwidth Concept (ABC) detector algorithm to generate these outputs. DEF at this time has two primary applications. The first application protects the payload of a file from unauthorized tampering or distribution. The second identifies an arbitrary segment of data by comparing results to the fileprints generated by running the algorithm on "truth" files of known types. A demonstration application was written in the Java programming language, which consists of graphical and textual modes of operation both of which are capable of producing fileprints on one or more target files. A general report is created for each file that is processed. A visualization of the prints created is presented to the user and processed statistically. The tamper detection capabilities of the system are strong, and it is able to detect very small differences in data segments. Further precision would make this feature more useful and open it up to more applications of the technology.

**14. SUBJECT TERMS**

Detector and Extractor of Fileprints (DEF), JAVA, Periodogram-based process, Swing, Adjustable Bandwidth Concept (ABC)

**15. NUMBER OF PAGES** 16

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |

# Table of Contents

# List of Figures

# 1. Abstract

## 1.1. Executive Summary

The Detector and Extractor of Fileprints (DEF) algorithm is a technique for representing data sequences in a reproducible and compressed form. It combines a periodogram-based process with the Adjustable Bandwidth Concept (ABC) detector algorithm to generate these outputs. The outputs of this process have a number of properties but two of these properties stand out.

- The fileprints are reproducible from machine to machine and platform to platform.
- The fileprints of like file types hold significant similarities.

From these two properties of DEF file-prints bring about the two primary goals of this effort. The first goal is to protect the payload of a file from unauthorized tampering or distribution by creating an associated file-print to deliver with the original file for comparison purposes. Along the same lines it holds that even if a file is changed in size or renamed, the unchanged segments of the file can still be identified as being consistent. The applications of this technique could include tampering detection and copyright/intellectual property rights claim analysis.

The second goal was to be able to identify an arbitrary segment of data by comparing it to the DEF outputs run on "truth" files of known types. Upon analyzing a segment of data, the DEF outputs could be statistically compared with known prints of known types. The one with the least amount of error in comparison would then be known as the "detected" segment data type.

With these goals in mind a demonstration application was written in the Java programming language. Design goals beyond the implementation of DEF included cross-platform compatibility and intellectual property/code reusability. The application created consists of graphical and textual modes of operation both of which are capable of producing file-prints on one or more target files. The graphical interface takes the next step by providing the user with the ability to analyze the outputs of the DEF process.

A general report is created for each file that is processed. This report provides information on the file itself, the outputs created and how long the operation took to complete. A visualization of the prints created is presented to the user for demonstration and engineering purposes. After these prints are created they are processed statistically and compared to a truth database of previously processed files and an attempt is made to detect what the MIME type of the segment or file and the comparisons made are presented to the user. If the application sees a data source once, subsequent calls on the same data source give the user the opportunity to compare the two sets of DEF outputs for differences and similarities.

The tamper detection capabilities of the system are strong. It is able to detect very small differences in data segments. The caveat of our approach thus far is that the precision is only as good as the segment size being processed. Further precision would make this feature more useful and open it up to more applications of the technology.

The blind data identification capabilities are currently at approximately 60% efficiency, taking into account some file types detect better than others. Currently the application only processes the first stage file print as a source of statistics. In the future the other stages of processing should be added to this process. A Bayesian analysis technique, normalization of highly variant truth samples and other statistical improvements can also be applied to improve the performance of the algorithm in the future.

# 2. Introduction

## 2.1. DEF/ABC Technology

Dr. Andrew Noga of the Air Force Research Laboratory (AFRL) developed the DEF algorithm. The DEF algorithm provides a repeatable process by which reproducible condensed data-prints can be generated. Potential applications of the algorithm include alternatives to modern watermarking approaches, data tamper protection and blind data segment content identification.

- DEF - Detector and Extractor of File-Prints
- ABC - Adjustable Bandwidth Concept Detector

Operation of the DEF algorithm starts with data segment selection. This segment could be a portion of a file or any other stream of data. After the selection process is complete, a periodogram (discrete Fourier transform) is calculated for that data segment. The result of that transformation is filtered by the ABC detector, which generates a pre-specified number of detection maps. A threshold is applied to the detection maps and then the data is stored in an efficient binary format. This binary output is known as the file-print or data-print.

The following diagram gives a general overview of the DEF algorithm.

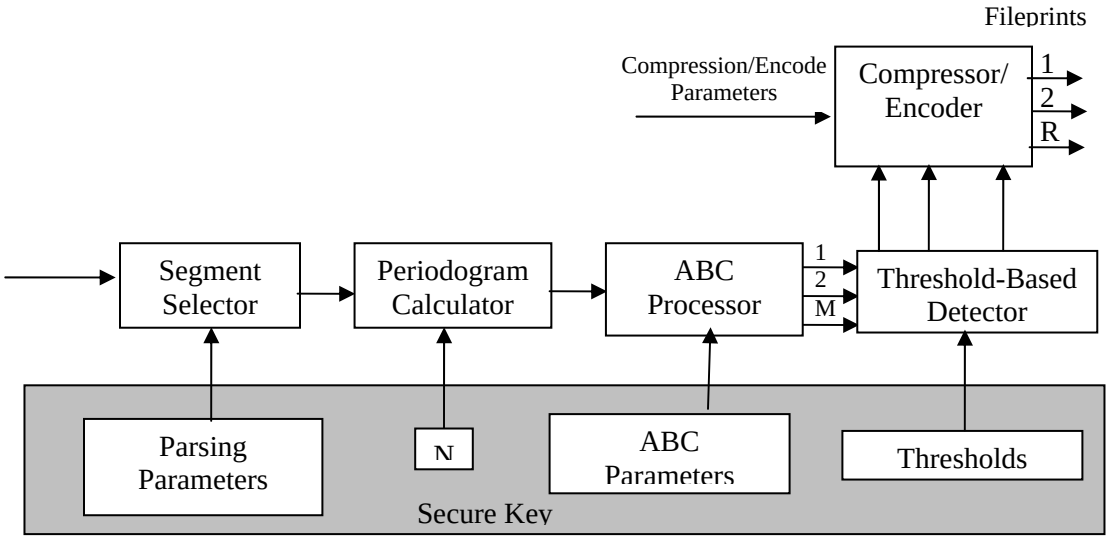


**Figure 1: The DEF Algorithm**

## 2.2. Java Technology

Java was chosen as the language for this effort primarily for its portability. The development platform consists of the following configuration:

- The Sun Java Developers Kit v1.4.2
- The NetBeans v3.5 IDE
- CVS Version Control System v1.11.1.3 (Build 57j)

All of these tools are freely available on the Internet and this environment is easily configured on most platforms including Windows, UNIX or Linux.

## 2.3. Other Technologies

To manage the blind data identification metadata and organize the previously processed data-prints, a Java relational database package was implemented as part of the demonstration. Hsqldb is a relational database engine written in Java, with a JDBC driver, supporting a rich subset of ANSI-92 SQL (BNF tree format). It offers a small, fast database engine that offers both in memory and disk based tables.

Another sub-technology to Java included in this project was the Jimi library, also developed and distributed by Sun Microsystems. This graphics conversion and processing library is used in DEF primarily in the context of image format encoding and decoding. This technology may be incorporated in future releases of Java.

# 3. Design and Implementation

## 3.1. Overview

There are three primary tasks associated with this portion of the DEF Java demonstration effort:

1. Implementation of the DEF and ABC algorithms and the user interface in the Java programming language.
2. File tamper detection using the DEF algorithm.
3. Blind file identification using the DEF algorithm.

Another goal of the development process was reusability of key technologies involved in the DEF process. These key technologies included the DEF and ABC algorithms, statistical techniques used in data-print processing and data format encoders and decoders. Efforts have been made to maintain separation of presentation and implementation.

## 3.2. Class Heirarchy

The DEF demonstration application is driven by the DEFUI class. This class contains the main method of the application which is specified in the applications JAR manifest. From this class the user can either run a set of inline routines for command line processing or start the Swing user interface to work in graphical mode.

Once in graphical mode, operation is driven by the applications "views". A view is a representation of the applications data pertaining to one or more functions of the DEF algorithm. Currently the application has four view types:

- DEFHTMLView
  - The base view for a processing session, generates the data-prints and the thresholds file and reports generated output, run time and other useful information back to the user.
- DEFTamperView
  - Show differences, both textually and graphically, between two sets of data-prints taken of the same target data. Used to find similarities and differences in data over time.

- DEFFIDView
  - Processes the file with some statistical techniques and compares the results with an internal database of truth data. Displays a graph representing error factors in comparison to the truth data set.
- printImageDump
  - Generates a graphical view of the detection maps, or data-prints, created by the DEF algorithm. Useful for side by side analysis and as a general purpose engineering and demonstration tool.

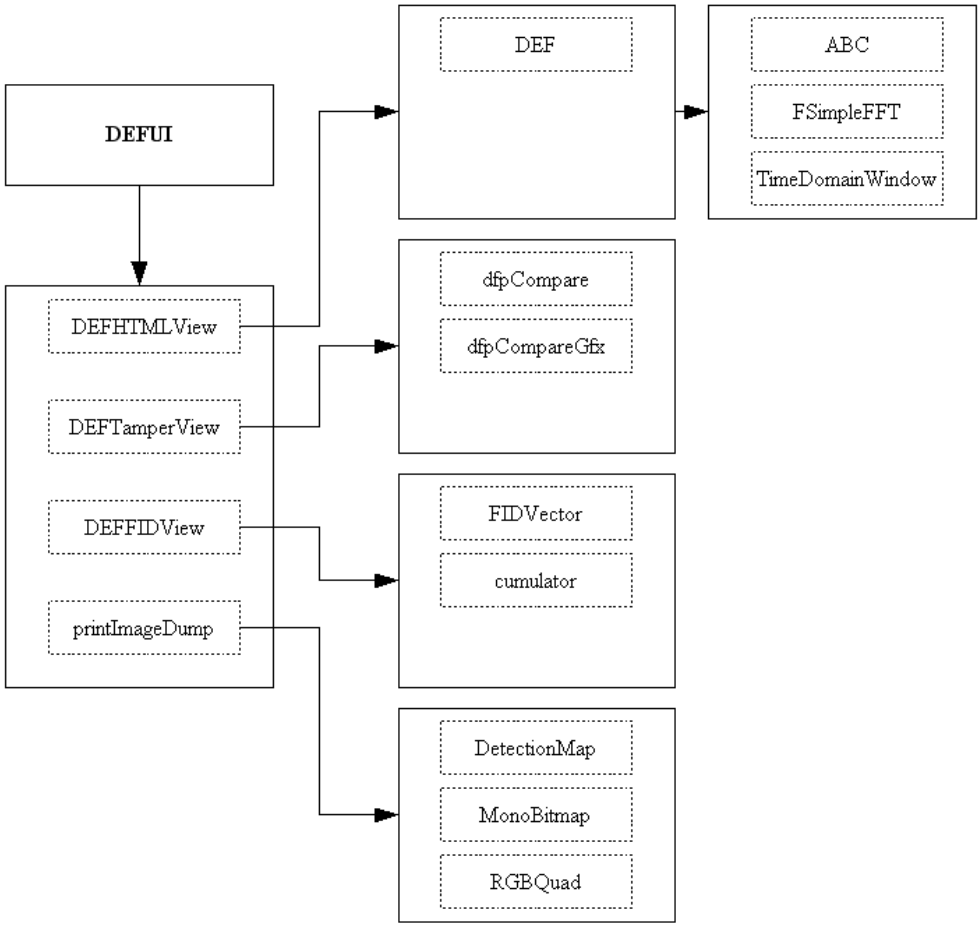


**Figure 2: Basic DEF Classes**

A number of utility classes not described here were also developed to assist in tasks related to mathematical operations, random number generation, graphical user interface and automation, all of which are suitable for reuse in this and other java applications.

## 3.3. User Interface

The user interface for this project was designed to try to meet a number of goals. An attempt was made to keep the functionality general enough to be useful as an engineering tool for further application development but compact enough to display potential uses of the application to interested parties.

When the application is first started the user is presented with the main application window. This window is made up of a main menu, a toolbar with some user preferences, a file navigation area and the general desktop where reports are displayed.

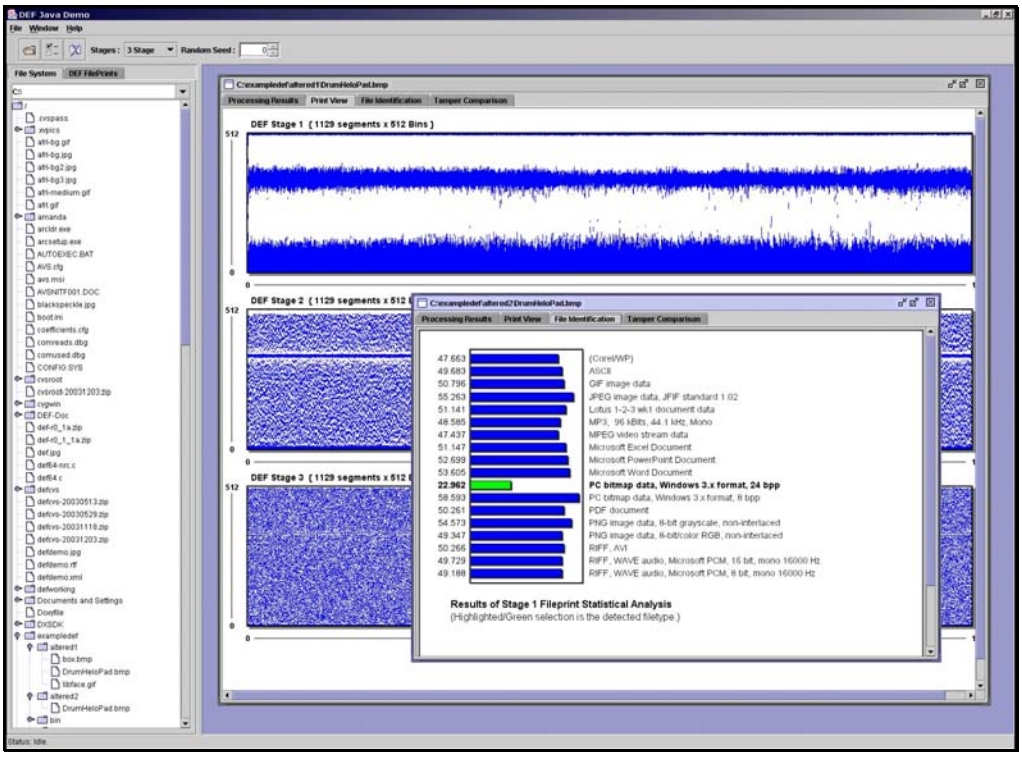


**Figure 3: The Main Application Window**

As the user begins to create data-prints they are indexed by file name and by the time of the printing and stored for later access. A listing of previously created prints, the number of stages included and the date and time the print was taken are displayed in the data-print database view. This view is found on the second tab in the left-most pane along with the file navigation tool.

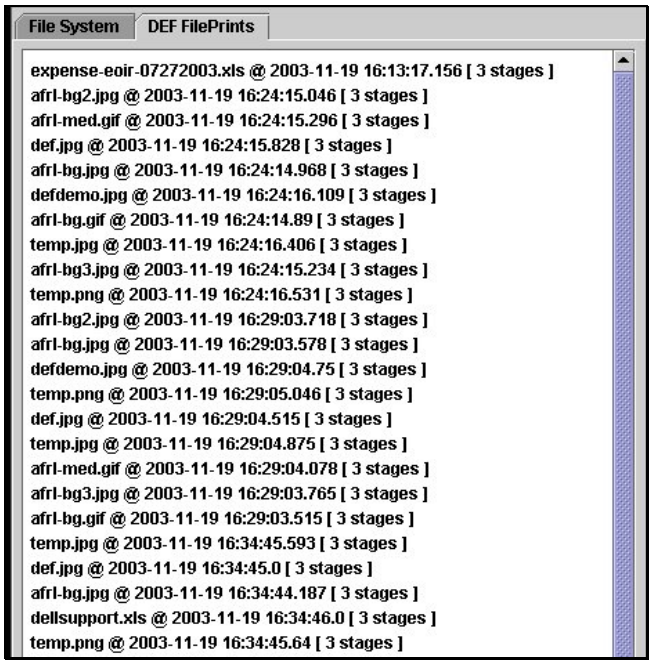


**Figure 4: The Data-Print Database**

Each time the user creates a print they are first presented with a general report of what has taken place. This report displays the file that was processed, the number of prints created and their locations, the time taken to create the prints and a ratio of data processed to time for benchmarking purposes.

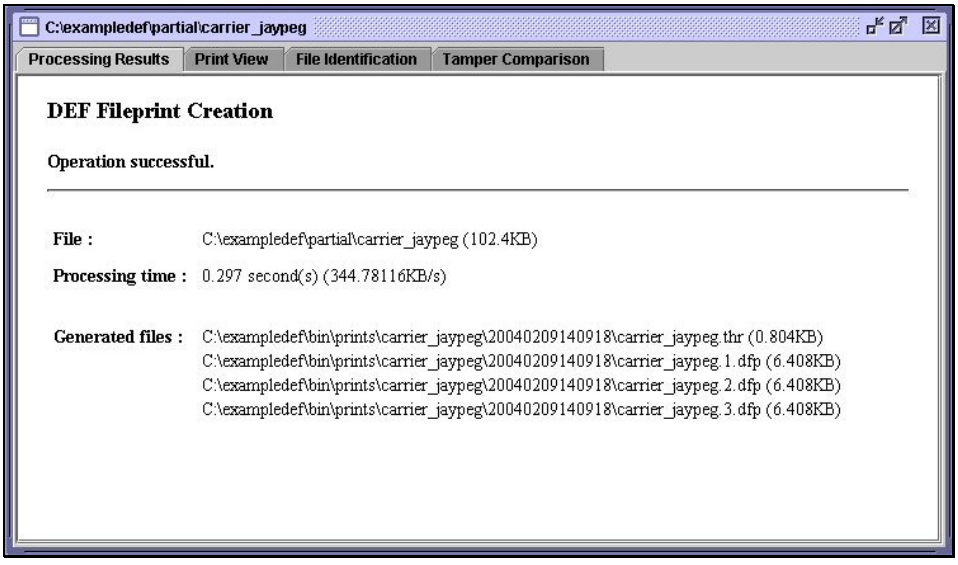


**Figure 5: The Main Report**

The next tab in the report window is the print view. In this view the user can actually view the prints created and look for interesting segments in the data. This view's primary purpose is as an engineering aid but is quite useful in a demonstration context.

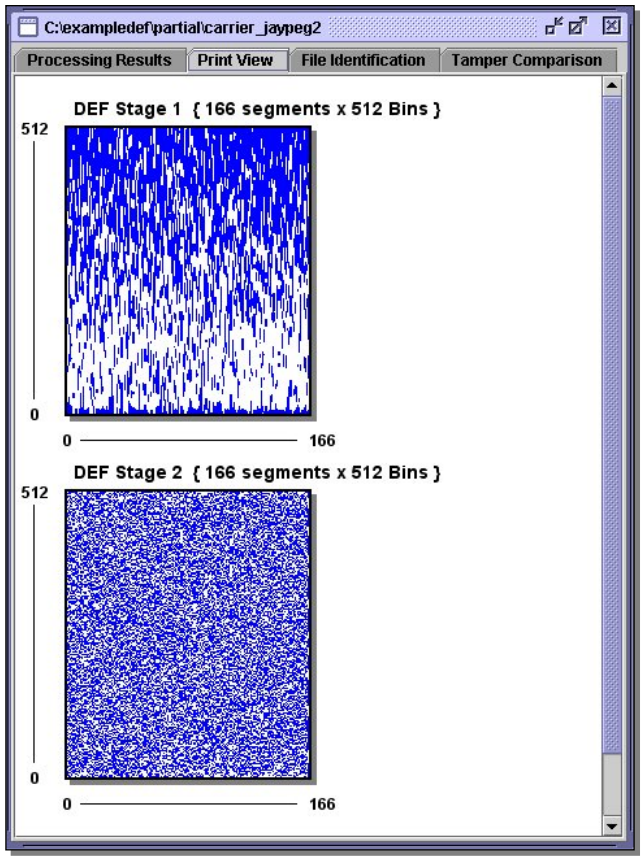


**Figure 6: The Print View**

After a print is created and stored in the database, subsequent runs on the same file give the user the ability to compare the two runs together and see if any changes have been made. Both a graphical representation and textual chunk-by-chunk comparison outputs are provided. A future use of this view may be to compare data to see what is the same, for example, in a copyright contention scenario.

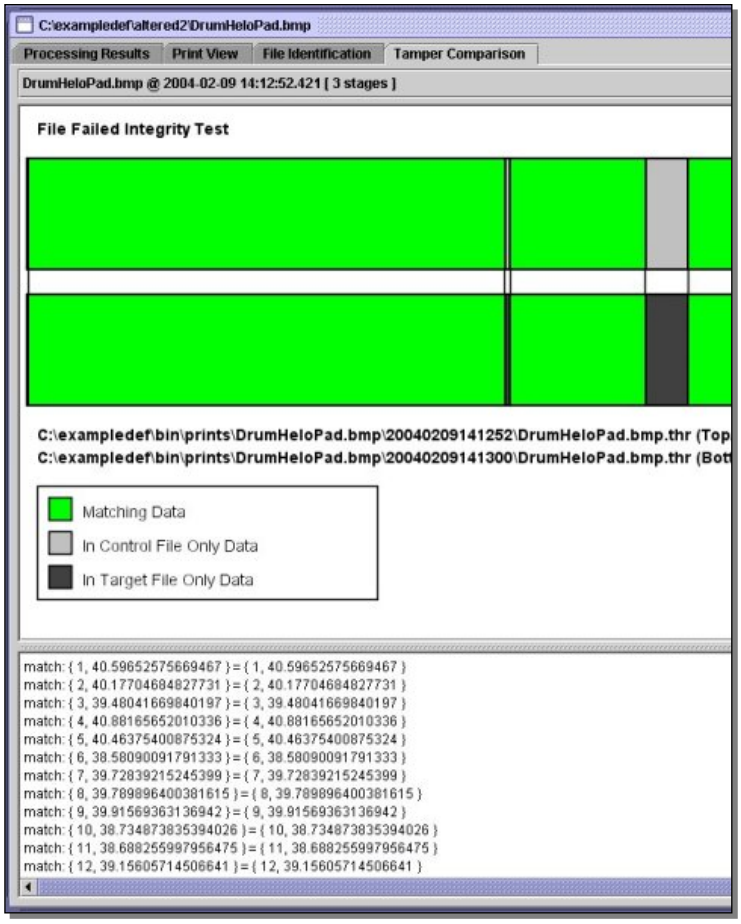


**Figure 7: The Tamper Detection View**

A truth database was constructed by creating prints for known file-types and storing statistical analysis of these outputs in an internal database. At runtime the users' data is analyzed in a similar fashion and then compared against this database. Rudimentary support has been implemented with interesting results. This view displays an error term for each known type of file in comparison to the target file. The green bar denotes the detected type.

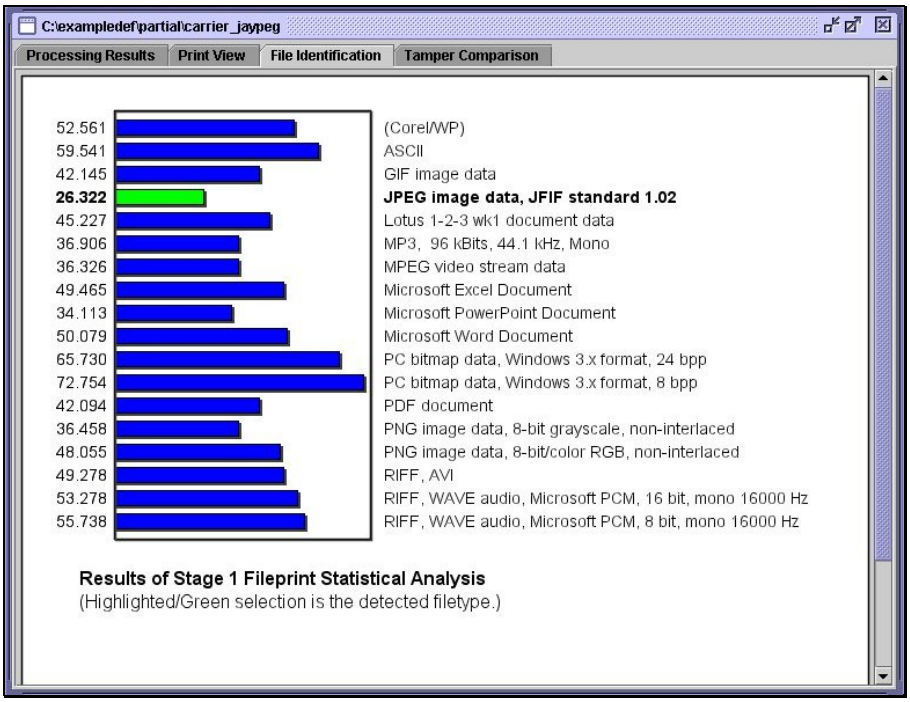


**Figure 8: The Blind File Identification View**

# 4. Conclusions and Future Efforts

## 4.1. Conclusions

This effort to date has provided very interesting results in both the tampering detection and blind identification areas. It has been successfully demonstrated as part of the Scientific Advisory Board (SAB) presentation for AFRL in November of 2003. With further development and refinement of the techniques started here, the DEF algorithm has a bright future as a potential part of numerous non-civilian and civilian applications.

## 4.2. Future Efforts

Future efforts should include the following:

- Improvement of the blind file detection capablilities.
- Improvement in the precision capabilities of the tampering detection algorithm.
- Better control for "batch-mode" processing.
- Refinement of the user interface.
- Other more application specific tasks.

# References

[1] U.S. patent 5,257,211 “Adjustable Bandwidth Concept Signal Energy Detector,” October 1993.

[2] Patent Pending; see http://www.uspto.gov/ Publication Number US/2004 0078574-A1.